2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

버퍼레벨을 이용한 멀티미디어 동기화 기법

성경상*, 이종찬**, 이근왕**, 이기성***
*숭실대학교 컴퓨터학과,**한국전자통신연구소,
***청운대학교 멀티미디어학과, ****호원대학교 컴퓨터학부

e-mail:actofgod@multi.ssu.ac.kr

Multimedia Synchronization Method Using Buffer Level

Kyung-Sang Sung*, Jong-Chan Lee**,
Kuen-Wang Lee***, Gi-Sung Lee****
*Dept of Computer, Soongsil University
**Electronics and Telecommunications Research Institute
***Dept of Multimedia, Soongsil University
****Dept of Computer Science, HoWon University

요약

QoS(Quality of Service)측면에서 멀티미디어 응용 서비스를 제공하기 위해서 멀티미디어 통신은 여러 가지 요구사항을 지원하여야 한다. 연속적으로 생성된 미디어의 시간적인 관계를 유지하면서 재생시킬 수 있는 동기화 메커니즘은 무엇보다도 중요한 문제이다. 전달되는 매체를 공동으로 사용하기 때문에 트래픽의 폭주시, 지역적으로 멀리 떨어져 있는 경우에는 지연의 현상이 발생하게된다. 본 논문에서는 지연에 의해 발생되는 지연지터에 대하여 완충역할을 할 수 있는 최소 버퍼크기를 제시하였다. 주문형멀티미디어서버(MOS)의 경우 각 미디어 별로 서버에 저장하여 서비스하는 연구가 현재 활발히 진행되고 있으며 이것은 네트워크의 부하를 줄일 수 있다는 장점을 가지고 있다. 또한 이러한 데이터의 경우 클라이언트에서 어느 정도의 지터를 예상하여 버퍼의 크기를 갖고 있어야 하는 문제는 아직도 큰 문제점으로 나타나고 있다. 본 논문은 버퍼의 크기 및 재생에 관한 동기화 시점을 조정함으로서 버퍼의 오버플로우나 언더플로우의 현상을 방지하고자 한다. 제안한 기법은 기존의 기법보다 향상된 동기화 기법임을 시뮬레이션을 통해 검증하였다.

1. 서 론

공통적으로 하나이상의 미디어의 시간적 순서를 유지하는 것을 멀티미디어 동기화라고 불린다[1]. 연속적인 미디어는 계속적인 데이터의 단위사이에 잘 정의된 시간적 관계에 의한 특성을 갖는다.

정보는 단지 미디어 내용이 시간적으로 연속적으로 표현될 때 전달된다. 비디오/오디오에 대하여 시간적 관계는 샘플링 비율에 의하여 감지된다. 단일 미디어내의 연속성 유지의 문제는 미디어내 동기화로 언급된다. 또한, 오디오와 비디오와 같은 관련된 미디어들의 미디어단위 사이의 시간적 관계도 존재한다. 이러한 시간제약의 유지는 미디어간 동기화라 불리어 진다.

미디어 동기화의 이러한 문제를 해결하기 위하여 우리는 미디어내 및 미디어간의 동기화 문제를 고려해야 한다.[2][3]

멀티미디어 동기화는 도착된 미디어가 시간적인 순서를 유지하기 위해서 초기의 버퍼 대기시간을 설정하고 동기화 시점을 주기로 각 미디어의 미디어간 동기화를 적용하는 것이다. 그러나 네트워크의 지연이나 지터에 의해 예상되지 않는 도착시점에서 각 미디어의 재생 정책을 이루어야 한다. 본 논문에서는 멀티미디어 시스템의 전송 지연에서 지터와 네트워크 상에서

미디어 손실이 존재하는 미디어 동기화에 관한 제에 대해 논의한다. 클라이언트는 네트워크 상에서 서로 다른 발신지로부터 도착된 미디어가 재생주기의 재생시간 안에서 버퍼의 용량과 흐름제어에 의한 동기화를 해결하여야 한다. 그러므로 본 논문에서는 버퍼의 점유율을 계산하여 피드백과 미디어의 상대시간 스탬프를 지정하여 하나의 구간에서 재생되어야 하는 각 미디어간의 재생주기를 결정하는 동기화 기법을 제안한다.

논문의 구성은 다음과 같다. Ⅱ장에서는 멀티미디어 동기화에 대한 관련 연구를 서술한다. Ⅲ장에서는 미디어처리기법에 관하여 서술한다. Ⅳ장에서는 재생처리기법에 관하여 서술하였으며 Ⅴ장에서는 제안한 멀티미디어 동기화 기법의 시뮬레이션 결과를 서술하고, Ⅵ장에서는 결론을 내리고 추후 연구 방향에 대하여 논의한다

2. 관련 연구

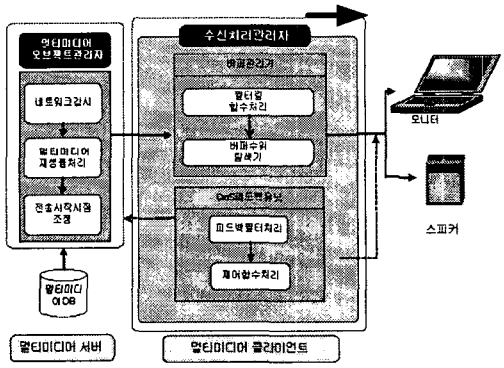
몇 가지 방법은 미디어내 동기화를 제안하였다. 첫 번째 방법은 플레이하기 위한 시간정보를 포함하는 프레임을 전송하는 것이다. 이것은 재생할때 클라이언트 쪽에서 스케줄을 하도록 한다. 두 번째 방법은 실제적인 도착시간과 기대되는 시간을 비교하여 네트웍 로드를 계산함으로써 도착된 프레임을 재생하거나 드롭시키는 것이다. Escobar은 이 기법을 위해 글로벌 클락을 제안하였다[7]. Little에 의한 세 번째 방법은 수신측에서 버퍼를 제공한다. 버퍼의 레벨을 체크함으로써 수신된 프레임은 재생되거나 네트워크 부하와 연관된 것은 폐기시킨다[8].

마지막 방법은 송신측에서 제어 기법을 제안 하였다. Rangan은 송신측이 전송 상태를 체크하여 프레임을 제어하는 피드백 기술을 제안하였다[9]. 수신측은 전송속도가 재생 되는 속도 보다 빠르거나 느릴 때 송신측에 피드백을 보내게 된다. 그런다음 송신측은 수신측으로부터 피드백에 관련된 어떤 프레임을 스킵하거나 같은 프레임을 보낸다. Chakrabarti's 방법은 피드백 기법이 수신지의 버퍼 레벨을 체크함으로써 생긴다는 것을 제외하고는 Rangan's의 방법과 같다[10]. 이 방법은 압축과 분해에 대한 시간을 고려하지 않는다.

3. 미디어처리 기법

3.1 전체시스템구조

본 논문에서는 각 미디어의 데이터가 멀티미디어서버에 저장된 형태를 가지고 있다. [그림 1]에서 보이는 바와 같이 미디어서버에는 오디오, 동영상, 에니메이션, 이미지, 텍스트 등을 서버에 저장하여 전송하는 방식이다.



[그림 1] 전체시스템구성도

3.2 버퍼관리자

버퍼 상태는 [그림 2]와 같이 정상레벨, 상위임계레벨, 상위제어레벨, 하위임계레벨, 하위제어레벨의 5가지의 상태를 갖게 된다. 정상 레벨에서는 처음에 미디어 데이터가 도착해서 미디어 데이터 버퍼가 정상적인 레벨에 있는 경우로 버퍼수위관리 유닛은 버퍼가 별 이상이 없으므로 계속 미디어 데이터를 전송 받도록 한다. 버퍼수위관리 유닛은 각 서버의 지터범위를 체크하고 정보를 저장한다. 상위임계·제어레벨에서는 네트워크가 평균 지연시간 보다 빨리 도착되어 네트워크의 상황이 좋아지는 경우이다. 버퍼가 상위임계레벨에 이르게 되면 버퍼가 오버플로우를 발생할 가능성이 높게 된다. 버퍼의 레벨이 상위제어레벨에 이르게 되면 버퍼는 오버플로우가 발생된 경우이다. 버퍼수위관리 유닛의 버퍼수위탐색기는 상위임계레벨에 버퍼의 수위가 옮겨가면 네트워크의 상태와 버퍼의 상태를 QoS피드백 유닛과 플레이아웃 유닛에게 통보하게 된다. 그리고 하위임계·제어레벨에서는 네트워크가 평균 지연시간 보다 늦게 도착되어 네트워크의 상황이 나빠지는 경우이다. 버퍼관리 유닛은 버퍼가 하위임계레벨에 이르게 되면 버퍼가 언더플로우를 발생할 가능성이 높게 된다. 버퍼의 레벨이 하위제어레벨에 이르게 되면 버퍼가 기아현상이 발생된 경우이다. 버퍼수위 유닛의 버퍼수위 탐색기는 상위임계레벨에 버퍼의 수위가 옮겨가면 네트워크의 상태와 버퍼의 상태를 QoS피드백 유닛과 플레이아웃 유닛에게 통보하게 된다.

[그림 2] 버퍼 상태 모드

3.2 필터링 함수

버퍼의 수위를 결정하는 방법에는 여러 가지가 있다. 현재의 버퍼수위로 결정하는 방법과 어떠한 제

어를 가하여 미래의 버퍼수위를 결정하는 방법 등이 있다. 본 논문에서는 버퍼 필터링함수를 적용하여 현재의 버퍼수위와 과거의 버퍼수위를 병합하여 수를 결정하는 방법을 사용하게 된다. 버퍼의 수위는 시간에 따라서 변하게 된다. 이러한 변화는 도착시간과 재생시간을 가지고 변경하게 된다. 버퍼의 수위가 변하는 상태를 함수 값으로 나타내게되면 아주 날카로운 형태를 가지고 변화를 일으키게 되는데 날카로운 함수를 부드러운 함수 값으로 변경하게 하는 것이 피드백 필터가 된다.

부드러운 함수 $S(q_{t,m})$는 시간에 따라 변하는 버퍼를 측정하는 방법을 제시한다. 시간 t에서 서브스트림 m을 위한 버퍼 레벨은 $q_{t,m}$에 의해 나타낸다. $q_{t,m}$의 값은 네트워크의 짧게 변화되는 지터에 의해서 버퍼의 수위가 연속적으로 변화되는 것을 예방하기 위한 부드러운 버퍼 레벨 $\bar{b}_{t,m}$를 계산하도록 필터링 함수로 전달된다.

$$\bar{b}_{t,m} = \alpha \cdot \bar{b}_{t-1,m} + (1-\alpha) \cdot q_{t,m}$$

$$(\ with \quad \alpha \in [0,1]) \ldots [식\ 1]$$

비동기 문제의 잠재적버퍼수위 반응은 $S(q_{t,m})$의 수행에 있어 강하게 의존된다. 좀더 느리게 $S(q_{t,m})$ 반응되면 마지막 재 동기화 단계는 더 많은 버퍼 공간인 상·하한임계레벨 버퍼 슬롯의 b_m^A가 가능한 많이 비동기화을 위해서 보상하게 될 것이다. 다른 면에서 $S(q_{t,m})$가 민감하게 반응하게 되고, 종종 재동기화가 불필요하게 행한다면 적은 버퍼 공간인 b_m^A가 충분하게 제공되어야 한다.

본 논문에서는 α를 0.6 이나 0.7의 값이 버퍼 요구와 재동기화에 반영되어 버퍼의 스무딩 버퍼레벨을 구하게 된다[54, 55]. 이 수치는 강하게 반응하지 않으면서 민감하게 재동기화의 수준을 변형하게 한다.

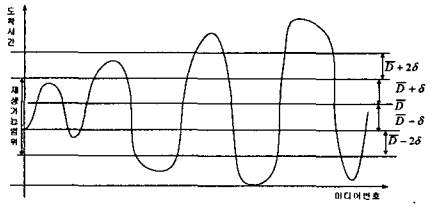
3.4. 제어 함수

제어함수는 상하임계·제어레벨에 버퍼의 수위가 존재할 때 안정상태로 어느 만큼 제어를 가할 것인가를 결정하는 함수이다. 피드백 필터링함수에서 구해진 부드러운 버퍼 레벨 $\bar{b}_{t,m}$ 는 네트워크의 상황과 버퍼함수를 취함으로써 제어함수 $C(\bar{b}_{t,m})$에게 전달된다. $\bar{b}_{t,m}$ 가 LW_m의 아래로 떨어지거나 UW_m를 초과한다면 각각 비동기를 발생하는 오버플로우나 기아현상의 위험을 발생시킨다. 이러한 현상이 일어난다면 재동기화나 적용 단계는 LW_m나 UW_m사이에서 $\bar{b}_{t,m}$ 를 정상상태로 이동시킨다.

$$RO_{t,m} = C(\bar{b}_{t,m}) = O_{t,m} + N_p \quad \ldots\ldots\ldots [식\ 2]$$

[식 2]에서 N_p는 [그림 3]에서 미디어 번호 m이 평가단계에서 얻어진 지연평균 $\bar{D}$ 의 일 분산의 범위를 구하여 네트워크의 지연을 추정하게 된다. 일 분산 범위는 68.3%의 데이터를 전송 받을 수 있는 범위가 된다. 이 범위는 [식 4-2]를 이용하여 옵셋을 결정하고 이 범위가 넘었을 경우는 이 분산의 범위가 되면 제어를 하게 된다. 네트워크가 느려지거나 빨라지는 경우이므로 제어를 수행하여 버퍼의 수위를 이동하게 하는 범위를 갖게 된다.

본 논문에서는 스케줄링 속도와 생산율을 미디어 서버에서 제어하는 알고리즘을 사용하게 된다. 클라이언트의 자원과 QoS을 위해서 서버의 자원을 이용한다는 장점을 갖게 된다. 클라이언트는 측정을 하면서 반응하는 부드러운 버퍼 레벨을 위하여 시간 RS^m를 위해 재동기화 단계에 머문다. 제어 함수는 버퍼관리자가 피드백처리기에게 버퍼레벨이 상·하임계레벨로 이동되었을 때 수행된다. 이때 피드백 필터링함수에서 구해진 값으로 수행된다. 이 때 얻어진 값을 이용하여 옵셋을 구하고 이 값으로 재동기화 단계 $C(\bar{b}_{t,m})$을 구하게 된다. 재동기화 단계가 수행되어 정상상태로 이동되면 재동기화 단계는 끝나게 된다. 그러나 계속해서 상·하임계레벨에 머물게 되면 재동기 단계를 다시 수행하여 옵셋값을 구하게 된다.



[그림 3] 네트워크의 상태

4. 시뮬레이션 결과

4.1 실험 환경

본 논문에서 제안한 기법의 실험을 위한 환경으

로는 IBM 호환 기종의 펜티엄 PC를 이용하였으며, 인터페이스 및 알고리즘은 Java 개발 킷 1.2.2로 구현하였고, 마이크로 소프트 MDB에 petrinet.mdb 화일로 저장된다.

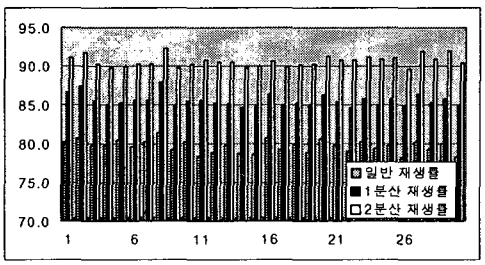
1Kbyte 오디오 데이터는 PCM 인코딩 기법에 의해서 인코딩되고 비디오 프레임의 해상도는 120 X 120을 사용했다. 초당 24프레임의 인코딩 작업을 하여 사용되어진 프레임이 된다. 송신측에서 어플리케이션은 125ms마다 오디오 디바이스로부터 오디오 패킷을 얻고 125ms동안에 어플리케이션은 운영체제의 런타임 프로세싱 오버헤드에 의해서 결정된 세 개 이하의 비디오 프레임을 비디오 그래버로부터 얻는다.

본 논문은 시뮬레이션 환경을 Ethernet상의 WAN 환경으로 가정하였다. 각각의 패킷에 대한 적절한 작업을 수행하기 위해서 실제 시뮬레이션에 사용된 정보는 포하송 분포로 산출하여 네트워크 지연 시간을 두가지 미디어에 똑같이 적용하였다.

4.2 실험 결과

본 절에서는 본 논문에서 제안하고 있는 제어함수를 적용하여 버퍼전략 및 피드백기법과 재생 기법의 재생시간과 손실 시간을 기존의 피드백기법과 비교 분석한다. 먼저 본 논문에서 버퍼전략적의 버퍼는 언더플로우나 오버플로우를 극복하는데 타 방법과의 비교를 통해 본 논문에서 제안한 방법의 우수성을 보였다. 각각의 미디어가 주미디어가 정상적으로 도착하였을 경우, 평균 지연은 100ms이고 편차는 20ms라고 가정한 후 실험하였다.

[그림 4]은 피드백에 의한 버퍼 점유율정책을 적용한 경우와 재생처리정책 만 적용되었을 경우와 피드백과 재생정책을 모두 사용하여 적용하였을 경우를 비교 평가한 결과이다. 본 논문에서 제안한 피드백과 재생처리를 모두 적용했을때, 제안된 전략의 재생율을 10번의 실험을 통해 얻어진 결과이다. 제안된 전략이 약 8% 이상의 재생율을 향상시켰다



[그림 4] 재생률 비교결과

5. 결 론

본 논문은 멀티미디어 시스템 및 서비스 제공에 있어 핵심적인 기술로 부각되는 동기화에 대한 동기화 기법을 제시하였다. 단일 서버를 이용한 멀티미디어 전송에서 사용된 버퍼의 크기를 분산 환경으로 확장함으로서 버퍼의 크기를 줄일 수 있었다. 또한 오디오와 텍스트 그리고 비디오 등의 데이터를 여러 서버에 저장함으로서 인기가 높은 미디어를 적절히 서비스 할 수 있었다. 또한 버퍼레벨을 사용한 기존의 기법에 네트워크 상태를 포함한 버퍼레벨 제어를 함으로서 버퍼의 기아 상태나 오버플로우 상태를 막을 수 있었다.

향후 연구 방향은 사용자와의 상호 작용을 고려한 정형화된 멀티미디어 기법을 만들고 이를 트랜스포트 프로토콜로 구현하여 시뮬레이션 하는 것이다. 그럼으로써 모든 멀티미디어 응용 프로그램에 적용 가능한 동기화 기법을 확립하는 것이다. 또한 최소 버퍼를 이용한 최적의 동기화 기법을 연구해야 할 것이며, 나아가 이동 통신에서의 동기화 기법을 연구해야 한다.

참고문헌

[1] G. Blakowski and R. Steinmetz, "A Media Synchronization Survey: Reference Model, Specification, and Case Studies," *IEEE Journal on selected Areas in Communications*, Vol.14, No.1, Jan. 1996.

[2] R. Steinmetz, "Synchronization Properties in Multimedia Systems," *IEEE Journal on selected Areas in Communications*, Vol. 8, No.3, Apr. 1990.

[3] N. U. Qazi, M. Woo, and A. Grafoor, " A Synchronization and communication model for distributed multimedia objects," *Proc. of ACM Multimedia*, 1993.

[4] E. Biersack, W. Geyer, and C. Bernhardt, "Intra- and Inter-Stream Synchronization for Stored Multimedia Streams," *IEEE Proc. of Multimedia'96*, pp. 372-381, 1996.

[5] D. L. Stone, and K. Jeffay, "An empitical study of delay jitter management policies," *Multimedia Systems/Springer-Verlag*, 1995.

[6] T. D. C. Little, and Arif Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services," *IEEE Journal on selected Areas in Communications*, Vol. 9, No.9, Dec. 1991.